RESUME SCORING USING NLP

PROJECT REPORT

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SUBMITTED BY

Meghna Lohani

16BCE1395

Aman Sharma

16BCE1224

(April 2019)

ABSTRACT

The objective of the project is to create a Resume Scoring algorithm using Natural Language Processing. The algorithm will parse resumes one by one and will create a Candidate Profile based on the skills mentioned in the resume.

A corpus is created using Sketch Engine, Wikipedia pages for various required skills (example : Machine Learning, Data Science, Software developer, Programming) are given as input,

Sketch Engine creates a corpus from the useful data available on the given links.

Word Embeddings are created from the corpus and these are used to match the skills in the candidate resume with the required skills.

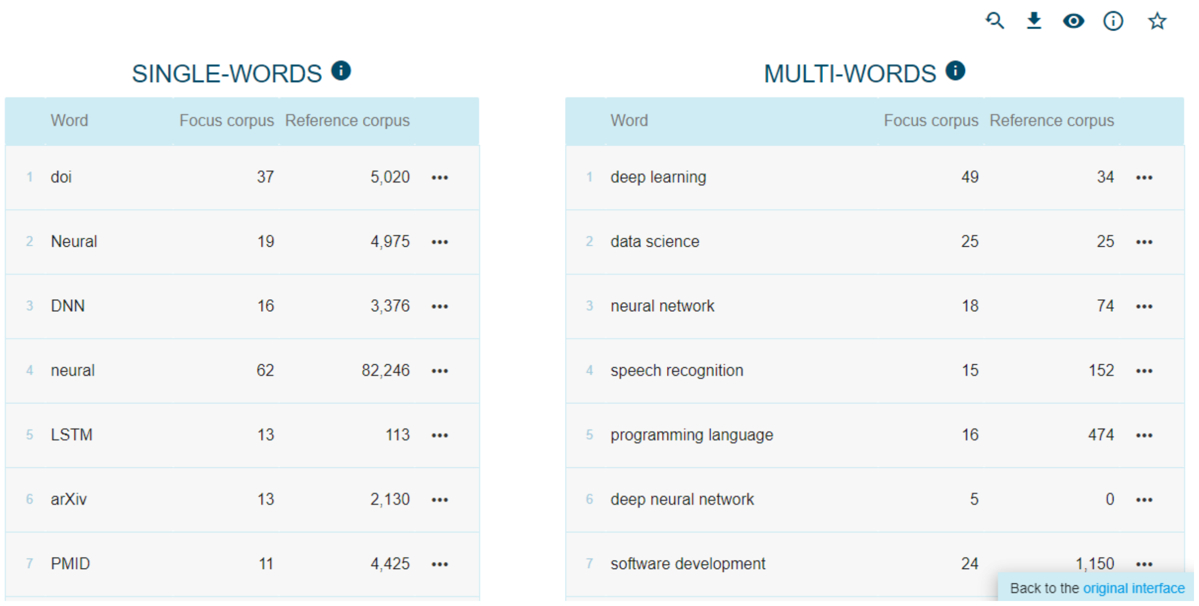
Finally, the candidate profile is built and plotted as a bar graph for better visualization.

Workflow

1. Creating corpus using Sketch Engine
2. Preprocessing
3. Removing common words that are not required
4. Creating phrases of words that frequently occur togther (bigrams)
5. Creating word embedding using Genism , Word2Vec
6. Testing the model
7. Extracting resumes (using PyPDF) and converting to string
8. Building candidate profile using model.most_similar(skills), where skills is an array of required skills and Creating a matcher using Spacy to match the wods in resume to most_similar(skills)
9. Counting words under each category to build a candidate profile.
10. Printing and visualizing the candidate profile

1. Creating corpus using Sketch Engine

Sketch Engine is an online tool to analyse how language works. It algorithms can analyse authentic texts of billion words to identify what is typical in a language, or rare or usual or emerging. It is used to create the corpus for our application since the available corpus did not contain the required words. Wikipedia pages are given as input and a text corpus is created from the useful information available on the given pages.



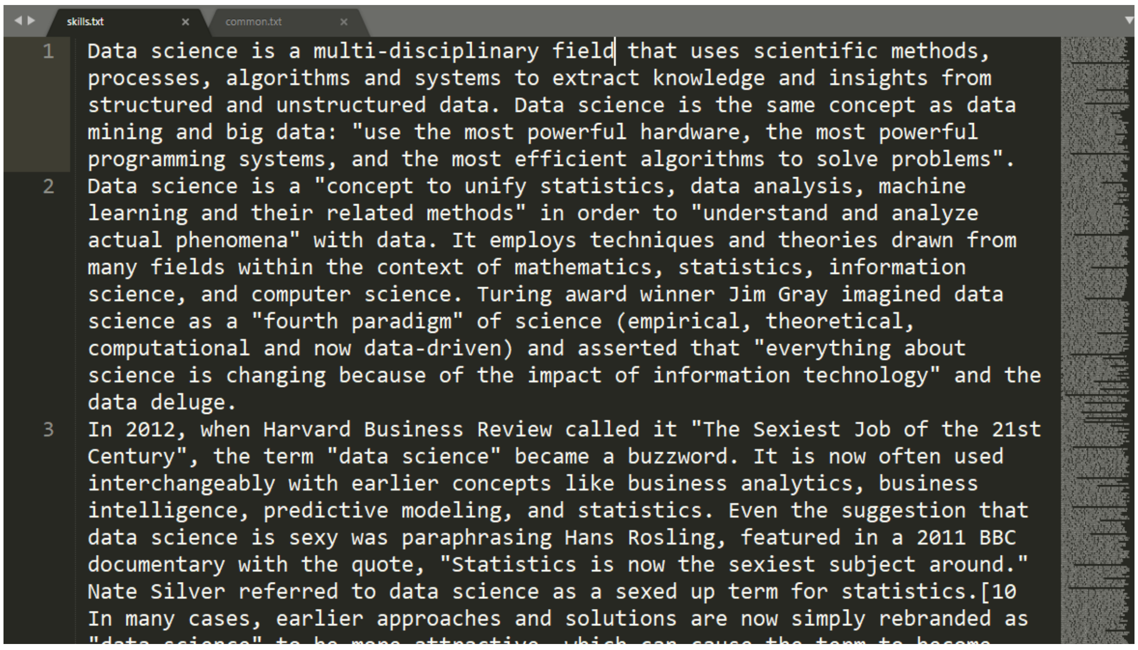
SINGLE-WORDS

	Word	Focus corpus	Reference corpus
1	doi	37	5,020
2	Neural	19	4,975
3	DNN	16	3,376
4	neural	62	82,246
5	LSTM	13	113
6	arXiv	13	2,130
7	PMID	11	4,425

MULTI-WORDS

	Word	Focus corpus	Reference corpus
1	deep learning	49	34
2	data science	25	25
3	neural network	18	74
4	speech recognition	15	152
5	programming language	16	474
6	deep neural network	5	0
7	software development	24	1,150

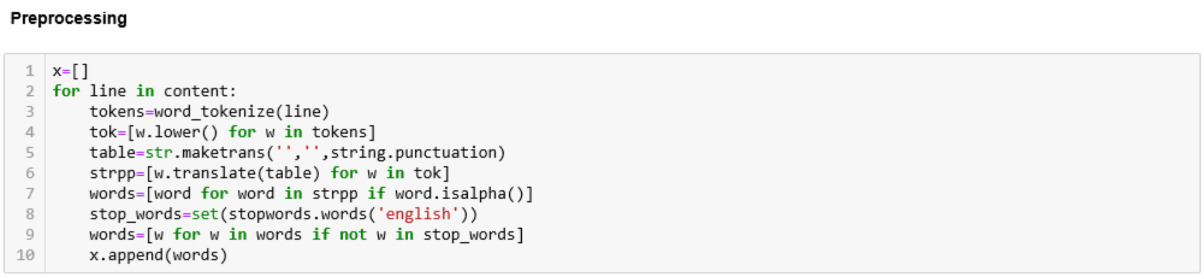
Corpus :



```
Data science is a multi-disciplinary field that uses scientific methods,
processes, algorithms and systems to extract knowledge and insights from
structured and unstructured data. Data science is the same concept as data
mining and big data: "use the most powerful hardware, the most powerful
programming systems, and the most efficient algorithms to solve problems".
Data science is a "concept to unify statistics, data analysis, machine
learning and their related methods" in order to "understand and analyze
actual phenomena" with data. It employs techniques and theories drawn from
many fields within the context of mathematics, statistics, information
science, and computer science. Turing award winner Jim Gray imagined data
science as a "fourth paradigm" of science (empirical, theoretical,
computational and now data-driven) and asserted that "everything about
science is changing because of the impact of information technology" and the
data deluge.
In 2012, when Harvard Business Review called it "The Sexiest Job of the 21st
Century", the term "data science" became a buzzword. It is now often used
interchangeably with earlier concepts like business analytics, business
intelligence, predictive modeling, and statistics. Even the suggestion that
data science is sexy was paraphrasing Hans Rosling, featured in a 2011 BBC
documentary with the quote, "Statistics is now the sexiest subject around."
Nate Silver referred to data science as a sexed up term for statistics.[10
In many cases, earlier approaches and solutions are now simply rebranded as
```

2. Preprocessing

Skills.txt is preprocessed before creating word emeddings, it involves removal of white space, stop words, digits etc and <p> tags.

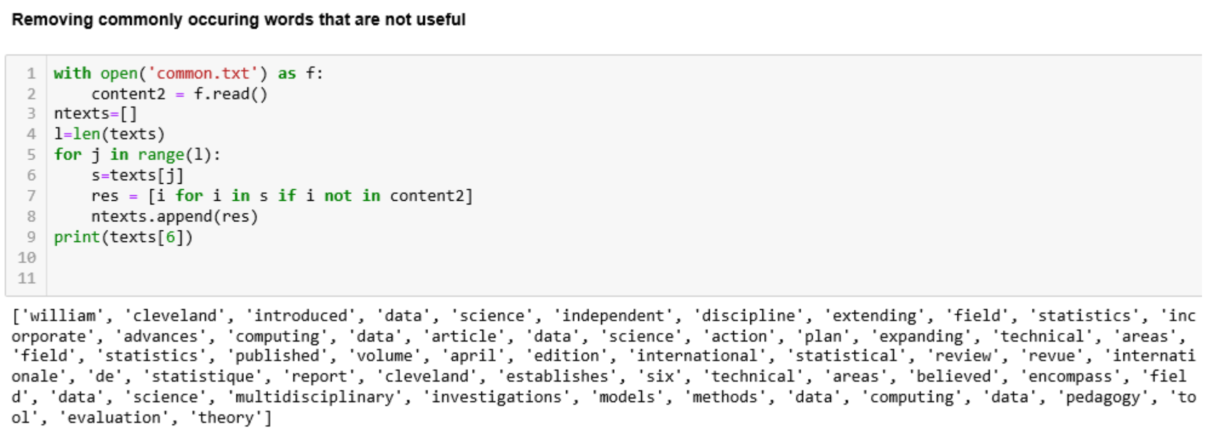


Preprocessing

```
x=[]
for line in content:
    tokens=word_tokenize(line)
    tok=[w.lower() for w in tokens]
    table=str.maketrans('','',string.punctuation)
    strpp=[w.translate(table) for w in tok]
    words=[word for word in strpp if word.isalpha()]
    stop_words=set(stopwords.words('english'))
    words=[w for w in words if not w in stop_words]
    x.append(words)
```

3. Removing common words that are not required

A text file is created named commons.txt which contains all the words that occur frequently and are not useful for technical skill matching (source : Wikipedia)

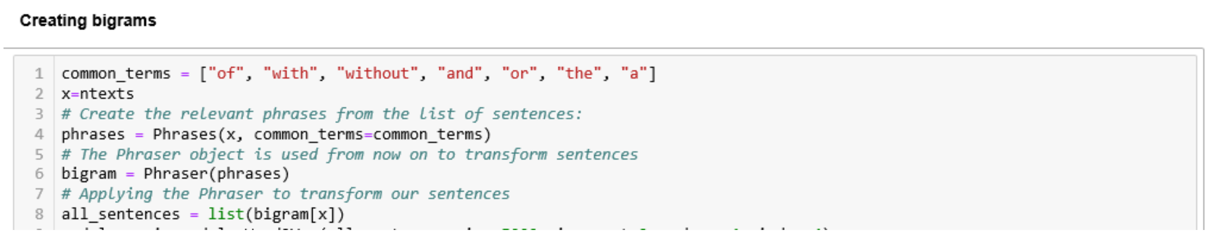

Removing commonly occuring words that are not useful

```
with open('common.txt') as f:
    content2 = f.read()
ntexts=[]
l=len(texts)
for j in range(l):
    s=texts[j]
    res = [i for i in s if i not in content2]
    ntexts.append(res)
print(texts[6])
```

```
['william', 'cleveland', 'introduced', 'data', 'science', 'independent', 'discipline', 'extending', 'field', 'statistics', 'inc
orporate', 'advances', 'computing', 'data', 'article', 'data', 'science', 'action', 'plan', 'expanding', 'technical', 'areas',
'field', 'statistics', 'published', 'volume', 'april', 'edition', 'international', 'statistical', 'review', 'revue', 'internati
onale', 'de', 'statistique', 'report', 'cleveland', 'establishes', 'six', 'technical', 'areas', 'believed', 'encompass', 'fiel
d', 'data', 'science', 'multidisciplinary', 'investigations', 'models', 'methods', 'data', 'computing', 'data', 'pedagogy', 'to
ol', 'evaluation', 'theory']
```

4. Creating phrases of words that frequently occur togther (bigrams)

Bigrams are those words which occur together almost all the time. Example : New York, Machine Learning. Bigrams are created using phrases.


Creating bigrams

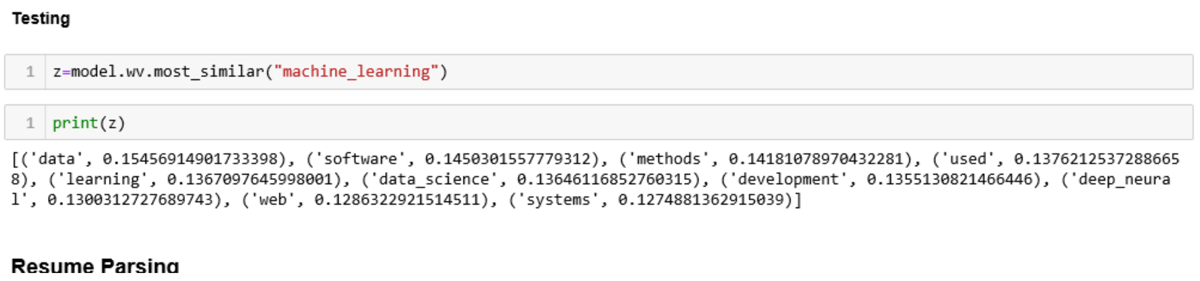
```
common_terms = ["of", "with", "without", "and", "or", "the", "a"]
x=ntexts
# Create the relevant phrases from the list of sentences:
phrases = Phrases(x, common_terms=common_terms)
# The Phraser object is used from now on to transform sentences
bigram = Phraser(phrases)
# Applying the Phraser to transform our sentences
all_sentences = list(bigram[x])
```

5. Creating word embedding using Genism , Word2Vec and building a word2vec model.

The Gensim library provides a simple API to the Google word2vec algorithm which is used to create word embeddings.
Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.

```
model=gensim.models.Word2Vec(all_sentences,size=5000,min_count=2,workers=4,window=4)
model.save("final.model")
wrds=list(model.wv.vocab)
print(len(wrds))
```

6. Testing the model.


Testing

```
z=model.wv.most_similar("machine_learning")
```

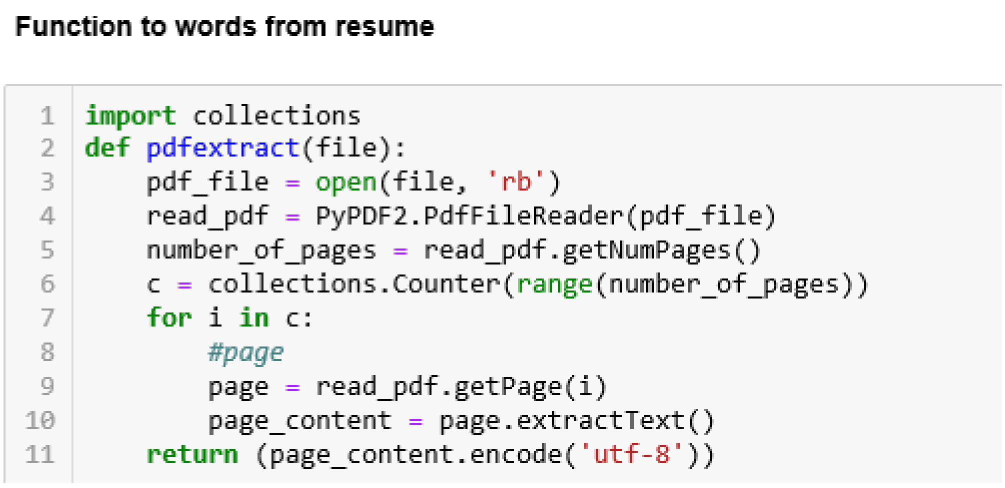
```
print(z)
```

```
[('data', 0.15456914901733398), ('software', 0.1450301557779312), ('methods', 0.14181078970432281), ('used', 0.1376212537288665
8), ('learning', 0.1367097645998001), ('data_science', 0.13646116852760315), ('development', 0.1355130821466446), ('deep_neura
l', 0.1300312727689743), ('web', 0.1286322921514511), ('systems', 0.1274881362915039)]
```

Resume Parsing

7. **Extracting resumes (using PyPDF) and converting to string**

Resumes are stored in a folder and extracted one by one using PyPDF library and returned as a sequence of strings. The string is preprocessed and is further processed to create candidate profiles.

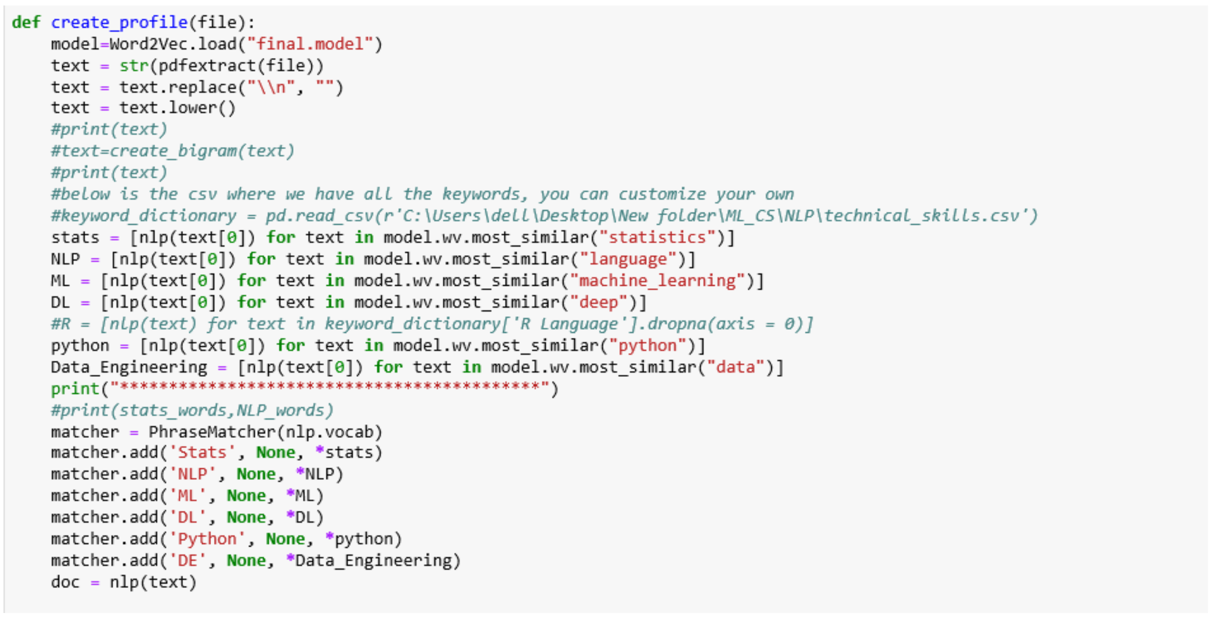


Function to words from resume

```python
import collections
def pdfextract(file):
    pdf_file = open(file, 'rb')
    read_pdf = PyPDF2.PdfFileReader(pdf_file)
    number_of_pages = read_pdf.getNumPages()
    c = collections.Counter(range(number_of_pages))
    for i in c:
        #page
        page = read_pdf.getPage(i)
        page_content = page.extractText()
    return (page_content.encode('utf-8'))
```

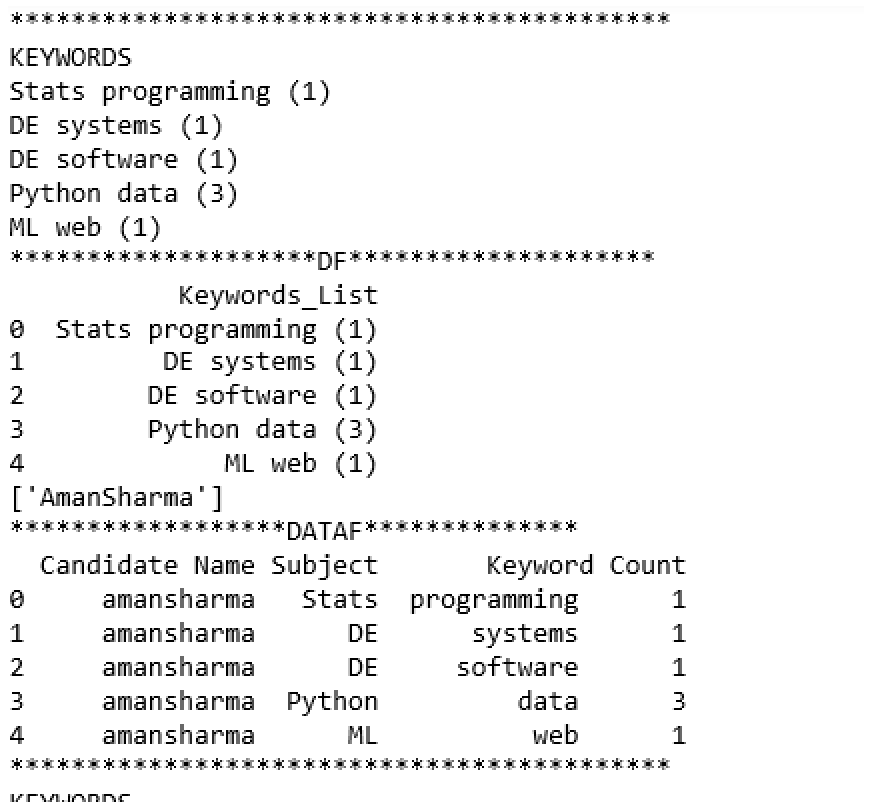
8. **Building candidate profile using model.most_similar(skills), where skills is an array of required skills and creating a matcher using Spacy to match the wods in resume to most_similar(skills)**

Spacy's Phrase matcher is used to match the array(obtained from word2vec) with the extracted text. It assigns each skill to a particular keyword and scores accordingly.



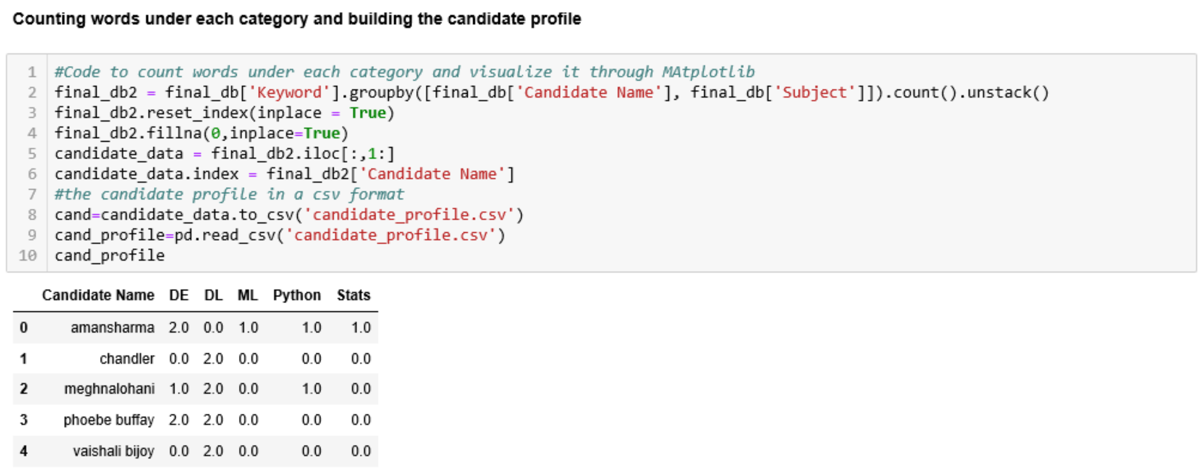
```python
def create_profile(file):
    model=Word2Vec.load("final.model")
    text = str(pdfextract(file))
    text = text.replace("\\n", "")
    text = text.lower()
    #print(text)
    #text=create_bigram(text)
    #print(text)
    #below is the csv where we have all the keywords, you can customize your own
    #keyword_dictionary = pd.read_csv(r'C:\Users\dell\Desktop\New folder\ML_CS\NLP\technical_skills.csv')
    stats = [nlp(text[0]) for text in model.wv.most_similar("statistics")]
    NLP = [nlp(text[0]) for text in model.wv.most_similar("language")]
    ML = [nlp(text[0]) for text in model.wv.most_similar("machine_learning")]
    DL = [nlp(text[0]) for text in model.wv.most_similar("deep")]
    #R = [nlp(text) for text in keyword_dictionary['R Language'].dropna(axis = 0)]
    python = [nlp(text[0]) for text in model.wv.most_similar("python")]
    Data_Engineering = [nlp(text[0]) for text in model.wv.most_similar("data")]
    print("*******************************************")
    #print(stats_words,NLP_words)
    matcher = PhraseMatcher(nlp.vocab)
    matcher.add('Stats', None, *stats)
    matcher.add('NLP', None, *NLP)
    matcher.add('ML', None, *ML)
    matcher.add('DL', None, *DL)
    matcher.add('Python', None, *python)
    matcher.add('DE', None, *Data_Engineering)
    doc = nlp(text)
```

Output



```
*********************************************
KEYWORDS
Stats programming (1)
DE systems (1)
DE software (1)
Python data (3)
ML web (1)
********************DF********************
           Keywords_List
0  Stats programming (1)
1         DE systems (1)
2        DE software (1)
3        Python data (3)
4             ML web (1)
['AmanSharma']
******************DATAF***************
  Candidate Name Subject      Keyword Count
0     amansharma   Stats  programming     1
1     amansharma      DE      systems     1
2     amansharma      DE     software     1
3     amansharma  Python         data     3
4     amansharma      ML          web     1
*********************************************
```

9. **Counting words under each category to build a candidate profile.**



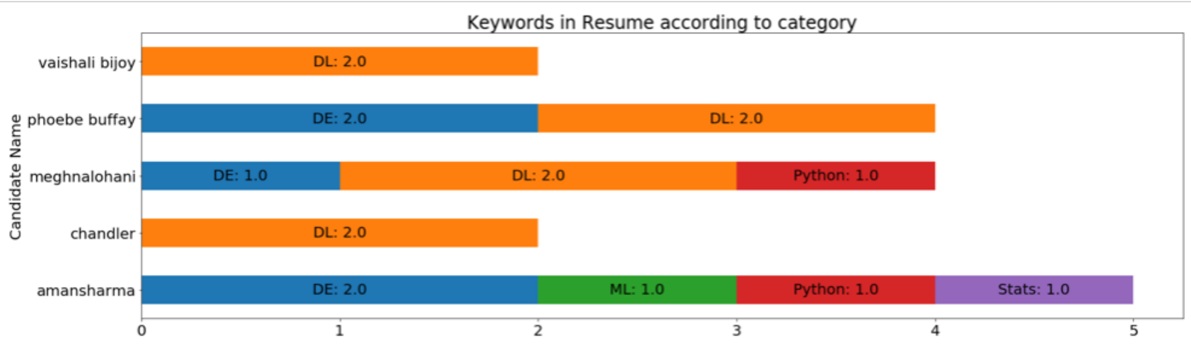
Counting words under each category and building the candidate profile

```python
#Code to count words under each category and visualize it through MAtplotlib
final_db2 = final_db['Keyword'].groupby([final_db['Candidate Name'], final_db['Subject']]).count().unstack()
final_db2.reset_index(inplace = True)
final_db2.fillna(0,inplace=True)
candidate_data = final_db2.iloc[:,1:]
candidate_data.index = final_db2['Candidate Name']
#the candidate profile in a csv format
cand=candidate_data.to_csv('candidate_profile.csv')
cand_profile=pd.read_csv('candidate_profile.csv')
cand_profile
```

	Candidate Name	DE	DL	ML	Python	Stats
0	amansharma	2.0	0.0	1.0	1.0	1.0
1	chandler	0.0	2.0	0.0	0.0	0.0
2	meghnalohani	1.0	2.0	0.0	1.0	0.0
3	phoebe buffay	2.0	2.0	0.0	0.0	0.0
4	vaishali bijoy	0.0	2.0	0.0	0.0	0.0

10. **Printing and visualizing the candidate profile**

The candidate profile is visualized through a bar graph using scores of each skill. So, the recruiter can easily interpret the candidate profile.



CONCLUSION

The application is able to match each skill in the candidate resume to a particular keyword (Machine learning, Programming) based on semantic distance between the words, i.e. using Word Embeddings. Matching is done by Spacy’s matcher and a score is created for each keyword. The final scores are visualized.

FUTURE SCOPE

With a better refined corpus, the model can be made more accurate. Due to the unavailability of the required corpus, we created a small corpus using Sketch Engine, therefore the vocabulary is limited.

REFERENCES

1. https://machinelearningmastery.com/develop-word-embeddings-python-gensim/

2. https://www.shanelynn.ie/word-embeddings-in-python-with-spacy-and-gensim/

3. https://app.sketchengine.eu/